Mobilizing Call of Duty: Bringing a Blockbuster Title to Android

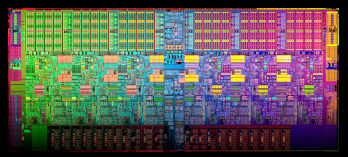
Reboot Develop Red 2019

Benjamin Mitchell
GameDev Engineer, Samsung Electronics

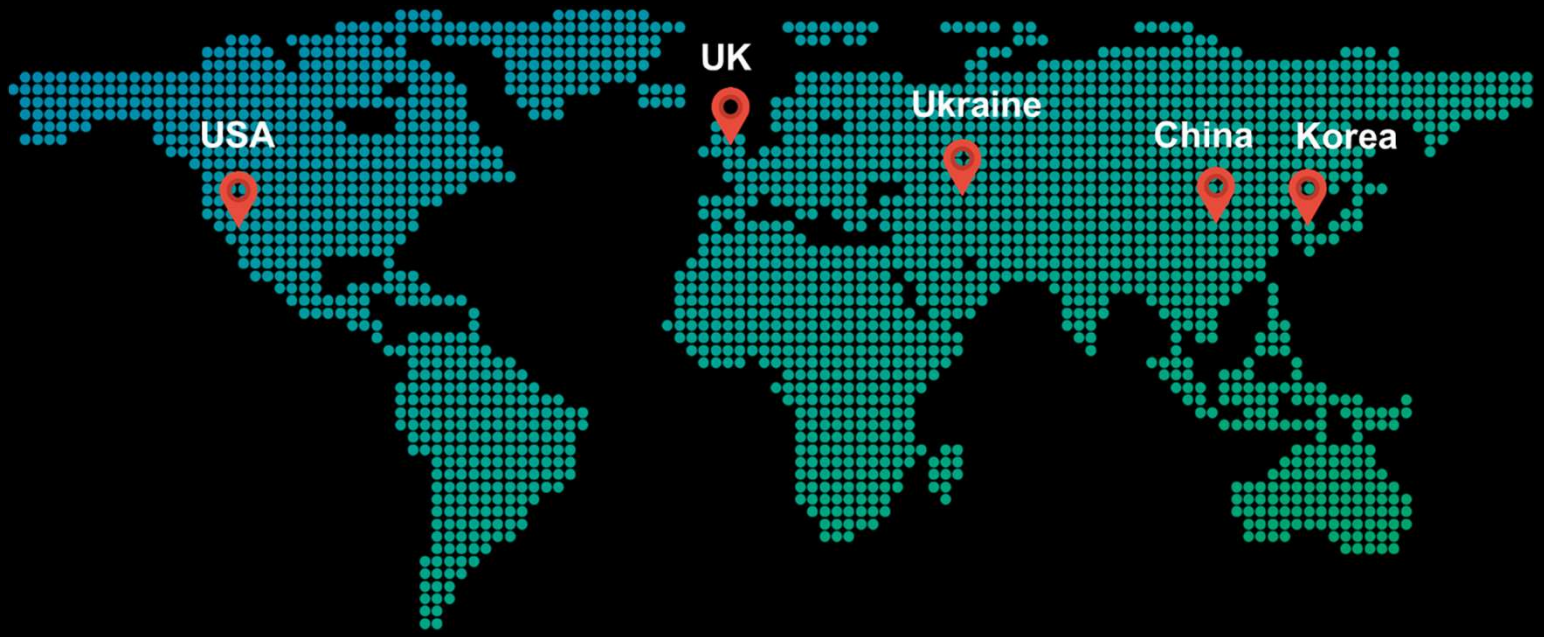
Galaxy GameDev

Vulkan®

64



Galaxy GameDev : since 2016



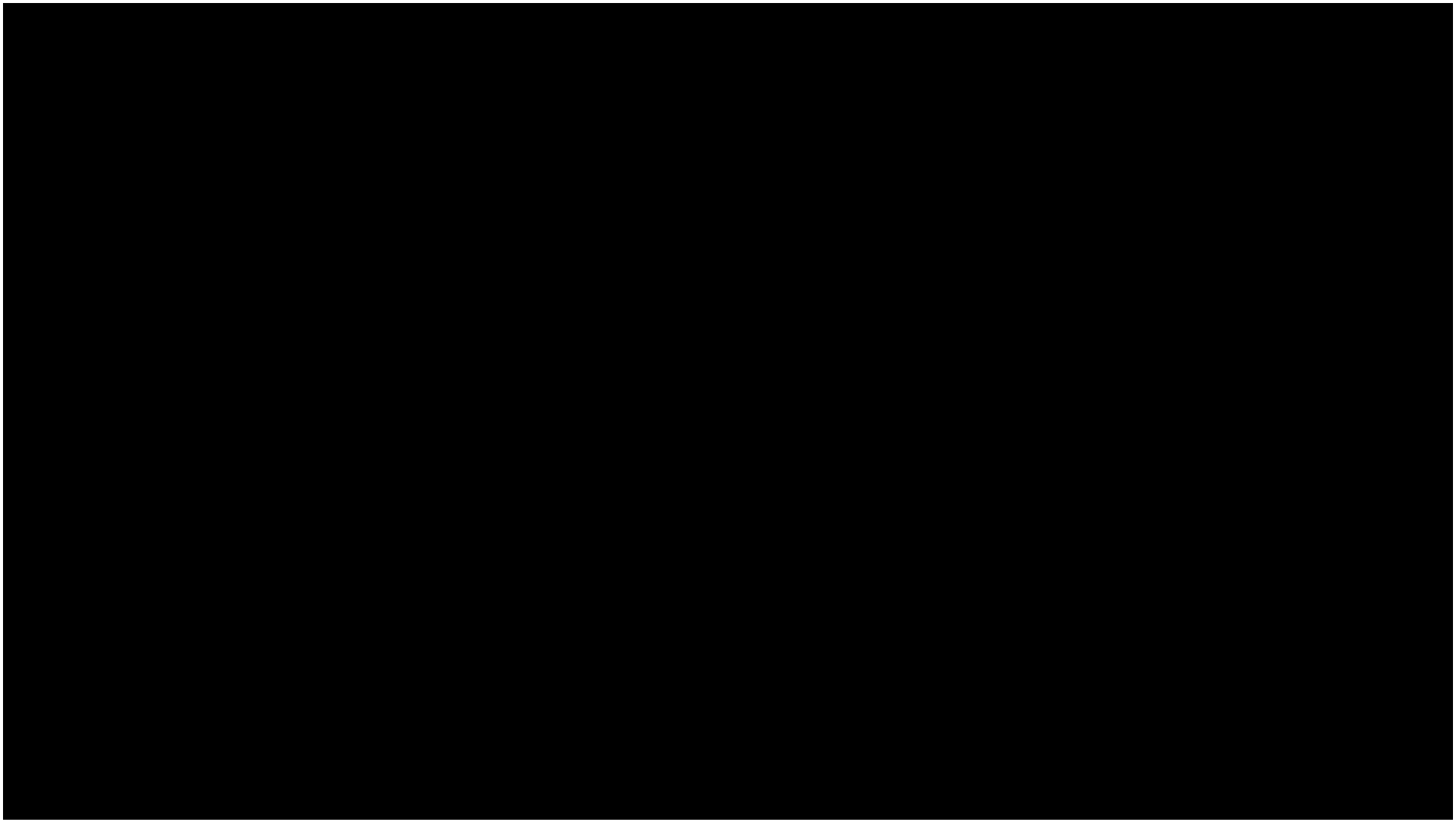
SAMSUNG

Galaxy GameDev


CALL OF DUTY
MOBILE
ACTIVISION
腾讯游戏
Tencent Games

Bringing Call of Duty to Mobile

- Call of Duty Mobile
- Adaptive Performance
- Vulkan Optimizations



Call of Duty Mobile

SAMSUNG

Galaxy GameDev

Call of Duty Mobile

SAMSUNG

Galaxy GameDev

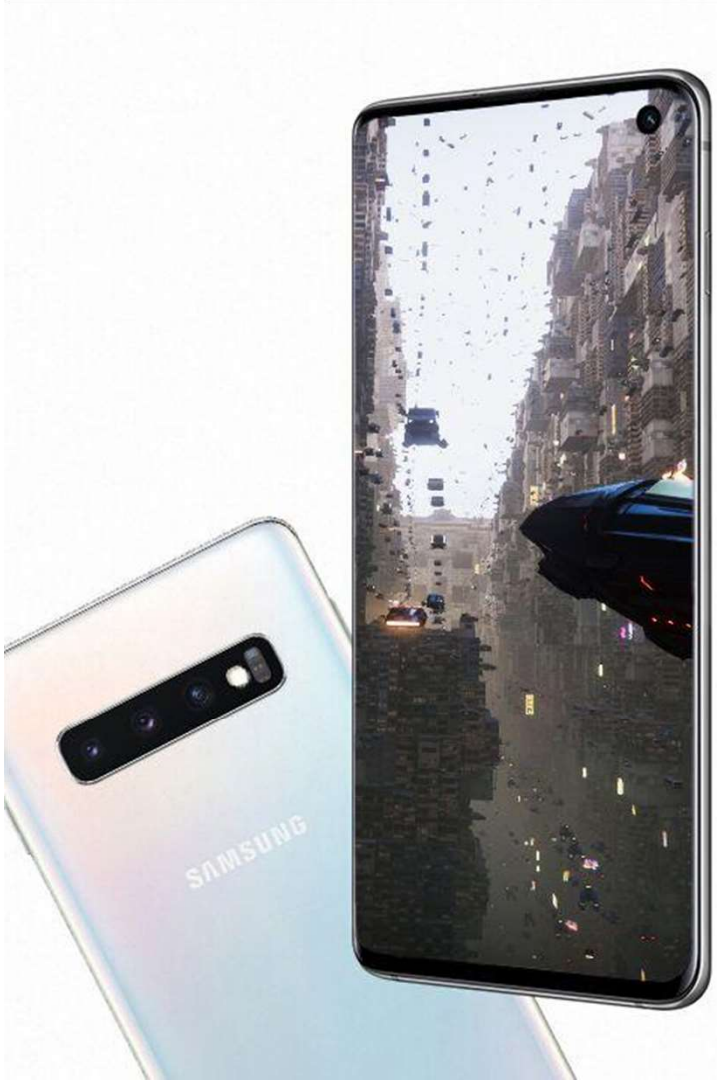
Call of Duty Mobile

SAMSUNG

Galaxy GameDev

Call of Duty Mobile

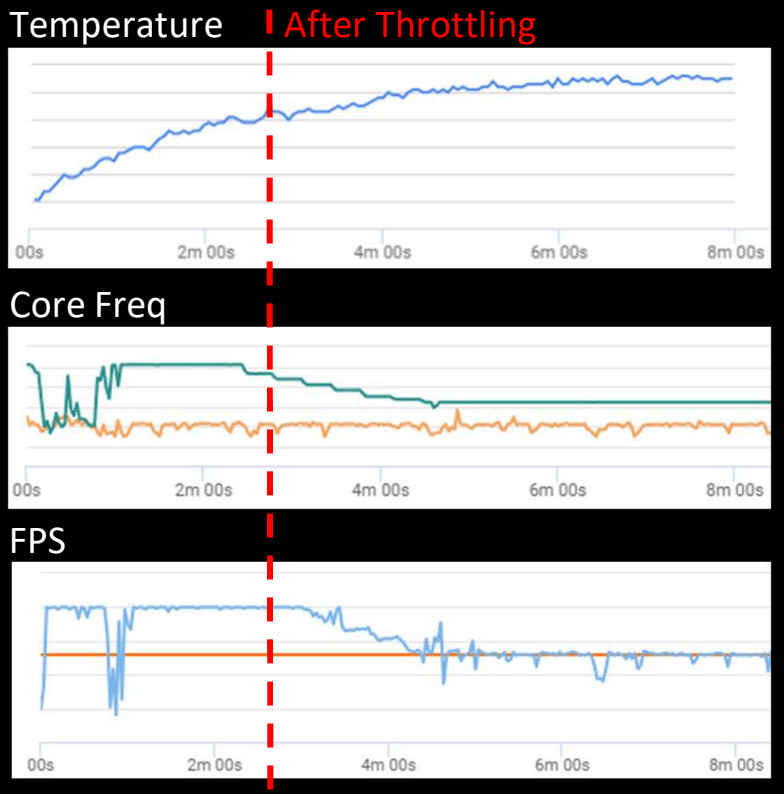
Call of Duty Mobile



Adaptive Performance

Mobile Limitation

- Temperature
  - No active cooling
  - Performance **Throttling**
- Power
  - Limitation of **Battery**
- Performance
  - Hardware **Fragmentation**
- Optimization is out of the Developer's control.



Adaptive Performance

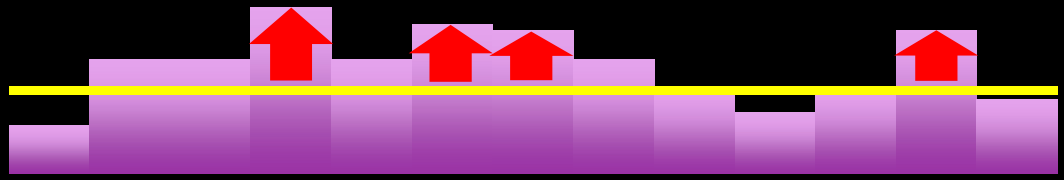
- Gather Data
  - Device Performance
  - Thermal Trends
- Adapt Performance
  - Adjust Quality Settings
  - Modify performance requirements

Galaxy Game SDK

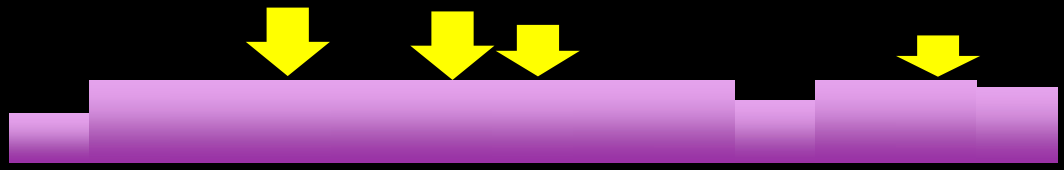
Power Manager

Power Budget Control

Default Core Frequencies

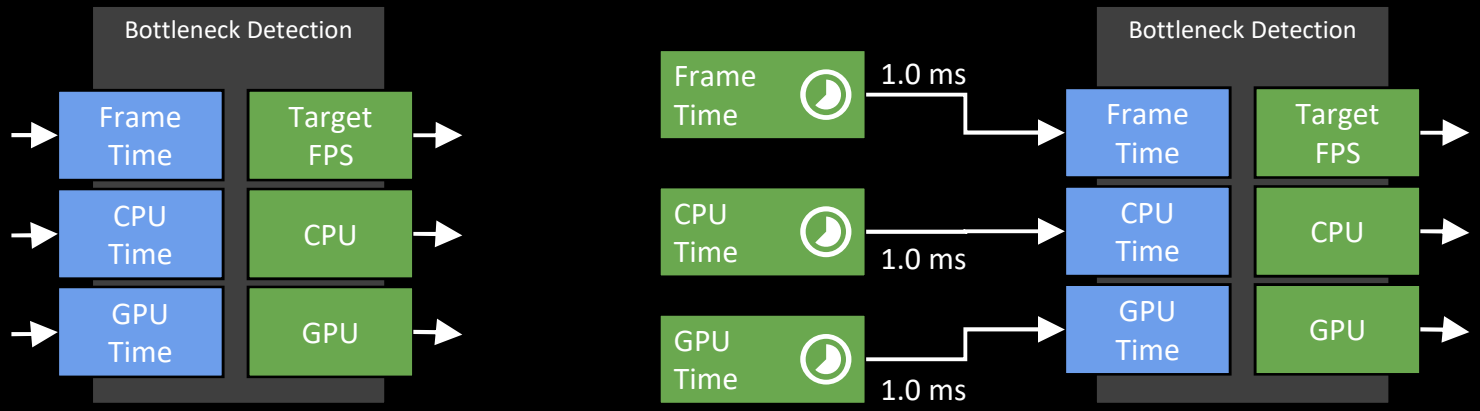


With Frequency Cap

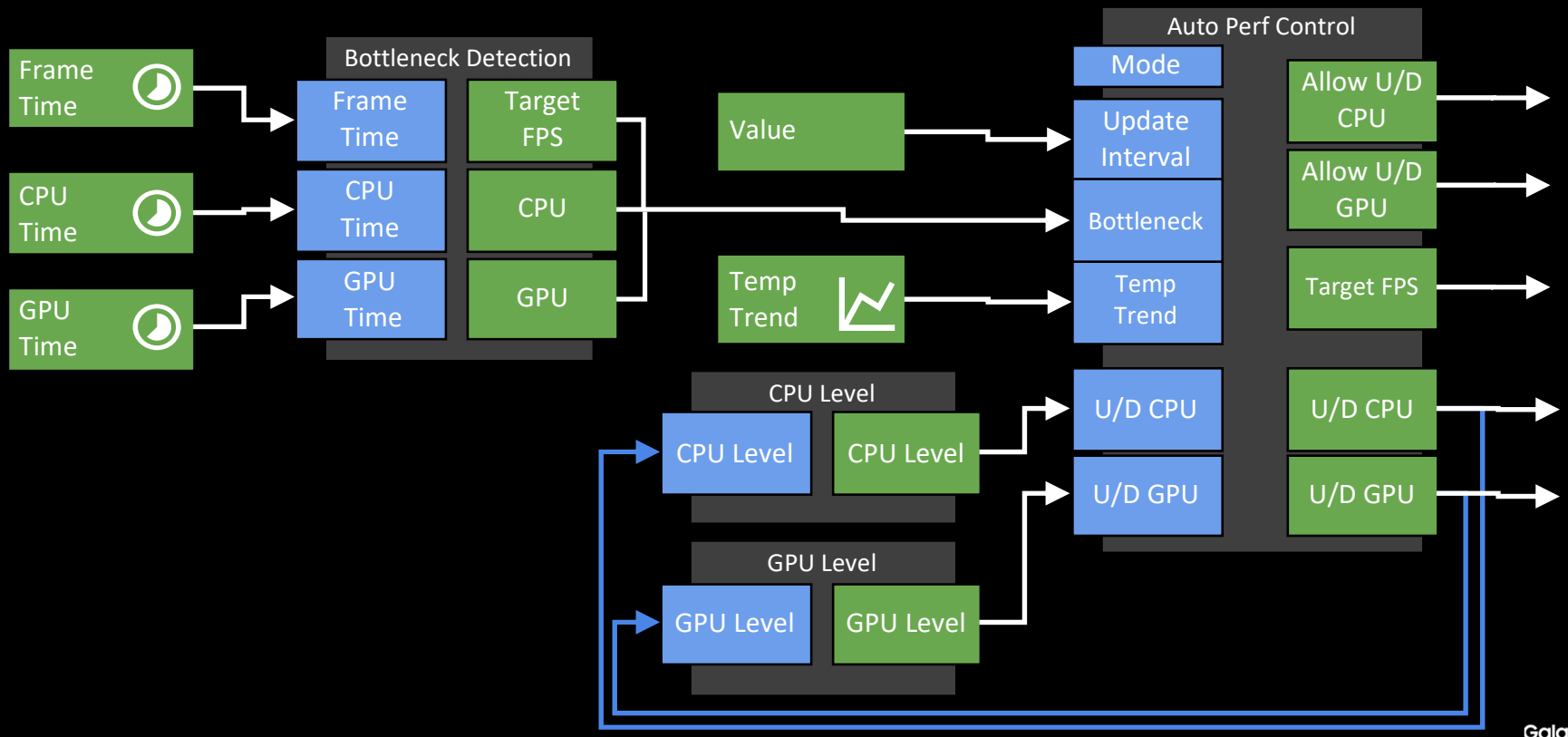


Power Manager

Frame Time Tracking



Default Power Manager

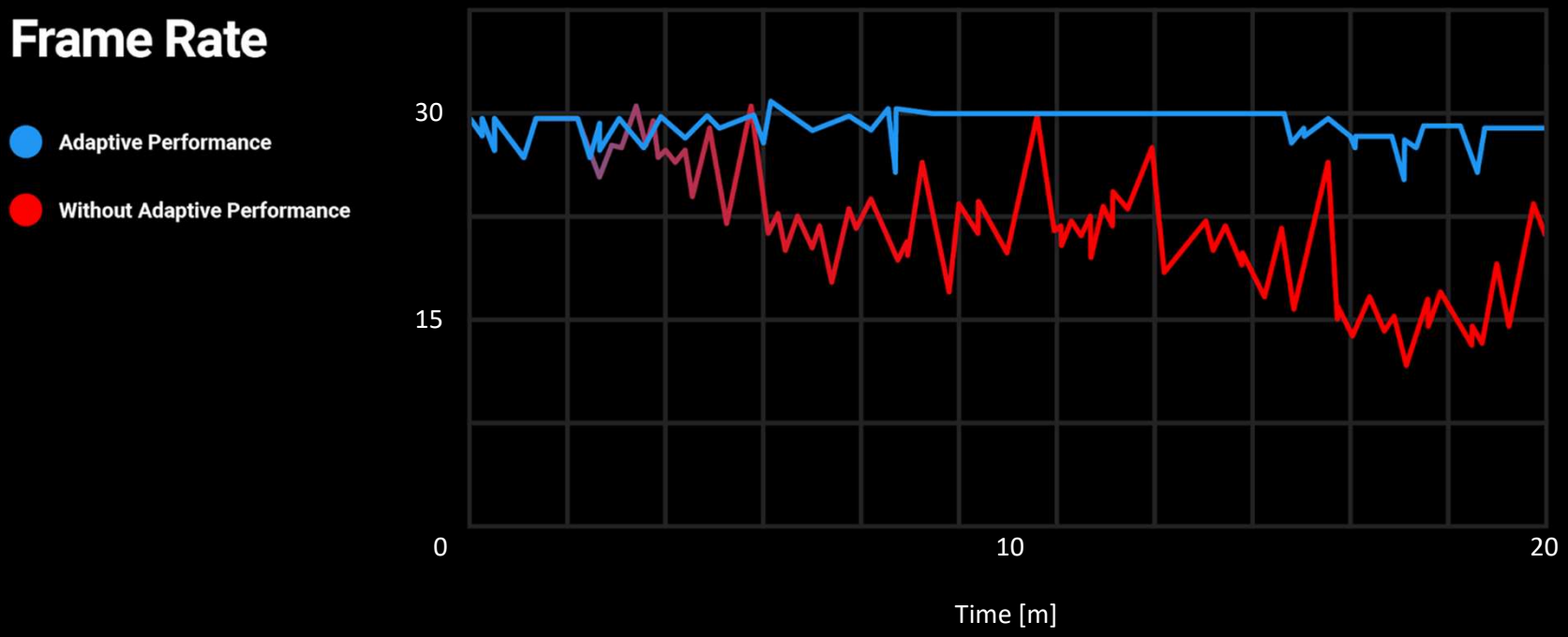



MEGACITY

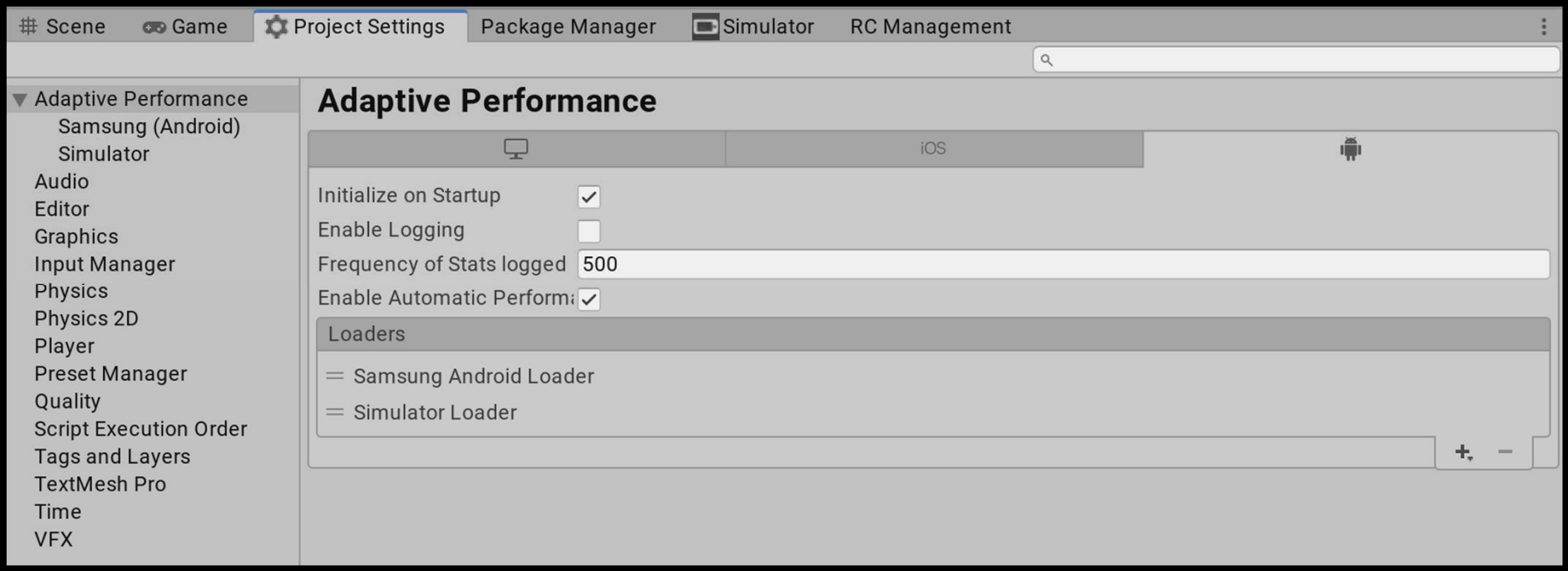
Adaptive Performance Quality Scaling

- Quality Scale Factors
  - Shadow/Rendering Distance
  - Texture LoD
  - Animation LoD
- Target Frame Rate

Power Manager + Dynamic Quality Scaling

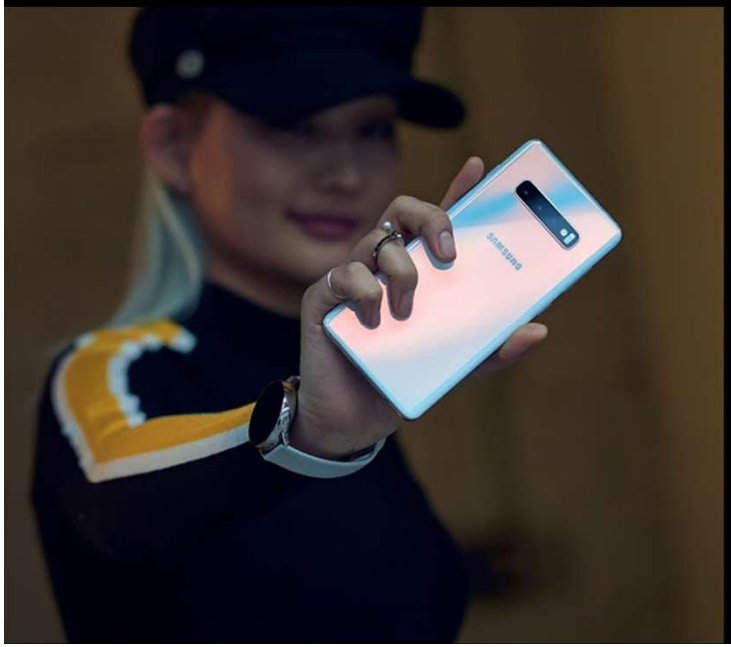


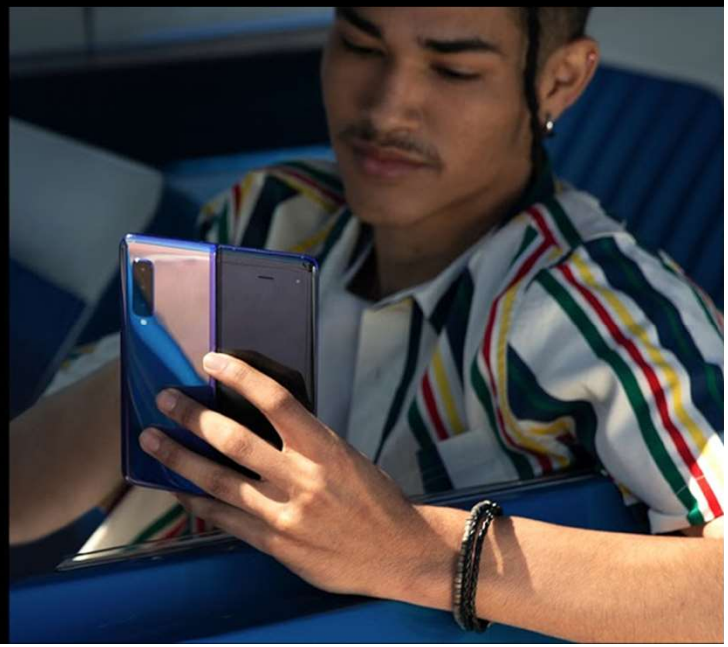
Adaptive Performance is now available



Adaptive Performance is now available

SAMSUNG | unity





Adaptive Performance in Call of Duty Mobile

SAMSUNG | unity

Adaptive Performance in Call of Duty Mobile

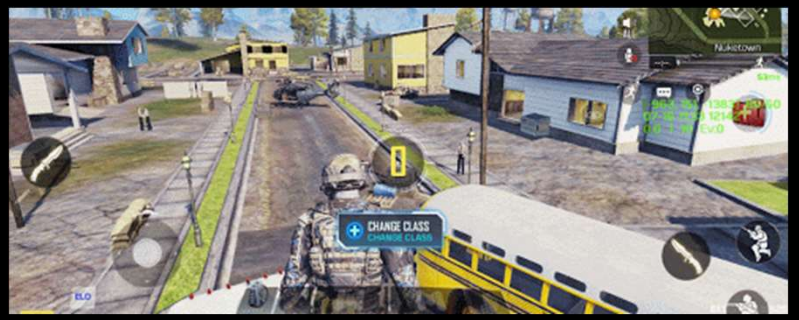
- Quality Scale Factors
  - Shadow Distance
  - Foliage LoD
  - Animation LoD
  - Target Frame Rate

- Power Management
  - Bottleneck Detection
  - CPU/GPU Power Budget Control

Adaptive Performance

Feature	Parameter	Scaling Impact
Shadow Distance	80 ~ 50	CPU 0~10% offloading
Foliage LOD	1 ~ 0.8	GPU 0~5% offloading
Animation LOD	0 ~ 2	CPU 0~3% offloading GPU 0~1% offloading
Target Frame Rate	57 ~ 60	CPU 0~5% offloading GPU 0~5% offloading

Shadow Distance(80 ~ 50)



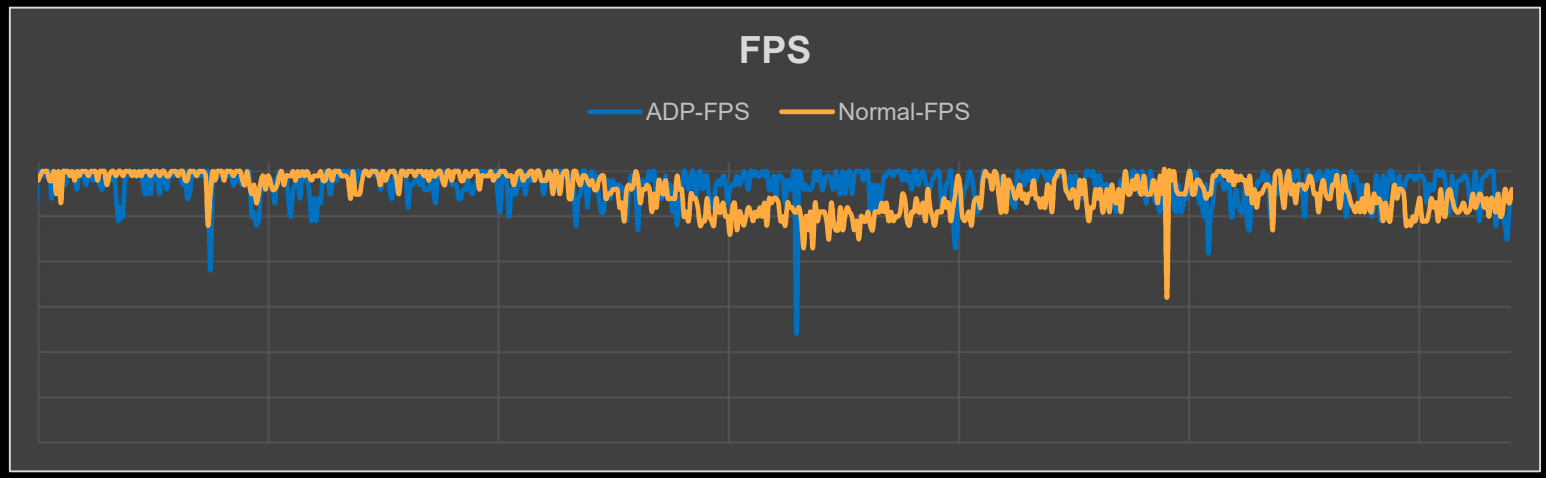
Foliage LOD(1.0 ~ 0.8)

Adaptive Performance

- Adaptive Performance

  vs Normal Condition

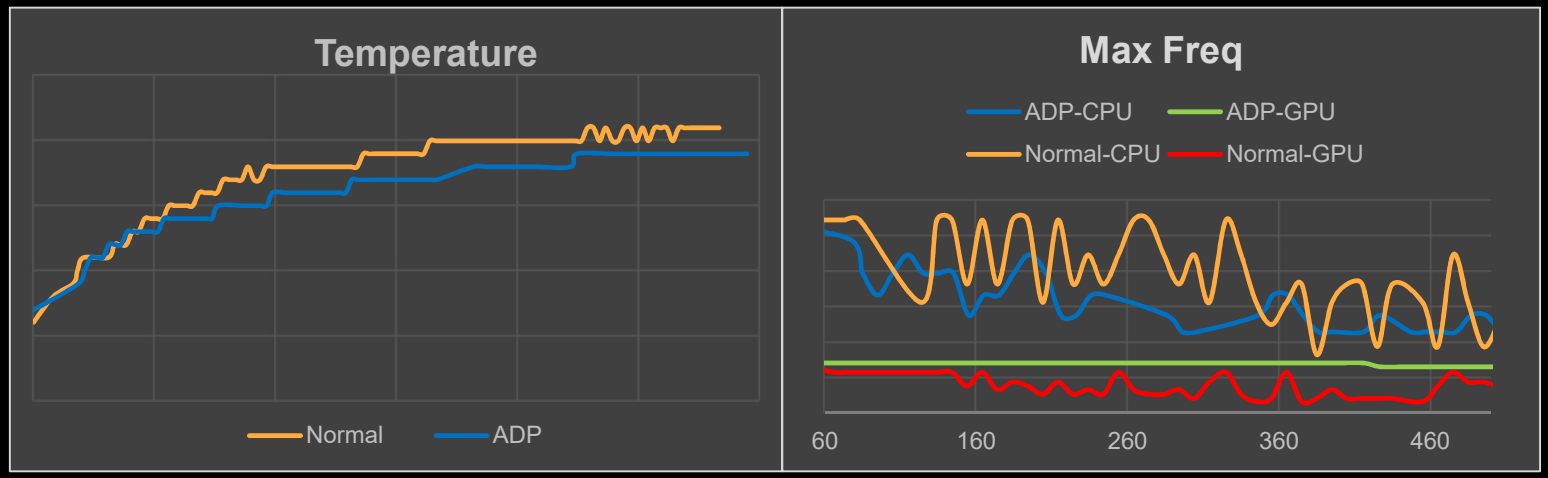
Feature	ADP	Normal
FPS	58	54
Stability (%)	100	100
CPU (%)	11.37	11.69
GPU (%)	80.08	87.85



Adaptive Performance

- Can save power and manage thermal better

Feature	ADP	Normal
FPS	58	54
Stability (%)	100	100
CPU (%)	11.37	11.69
GPU (%)	80.08	87.85




Vulkan™
KHRONOS GROUP
THE FUTURE OF GRAPHICS

General Vulkan Optimizations

- Pipeline Barriers
  - Ensure most optimal stage masks
  - Batch them!
- Load/Store Ops
  - You don’t always have to care!
  - LoadOpLoad and StoreOpStore
- Cache Stuff
  - PSOs
  - Descriptor Sets

General Vulkan Optimizations

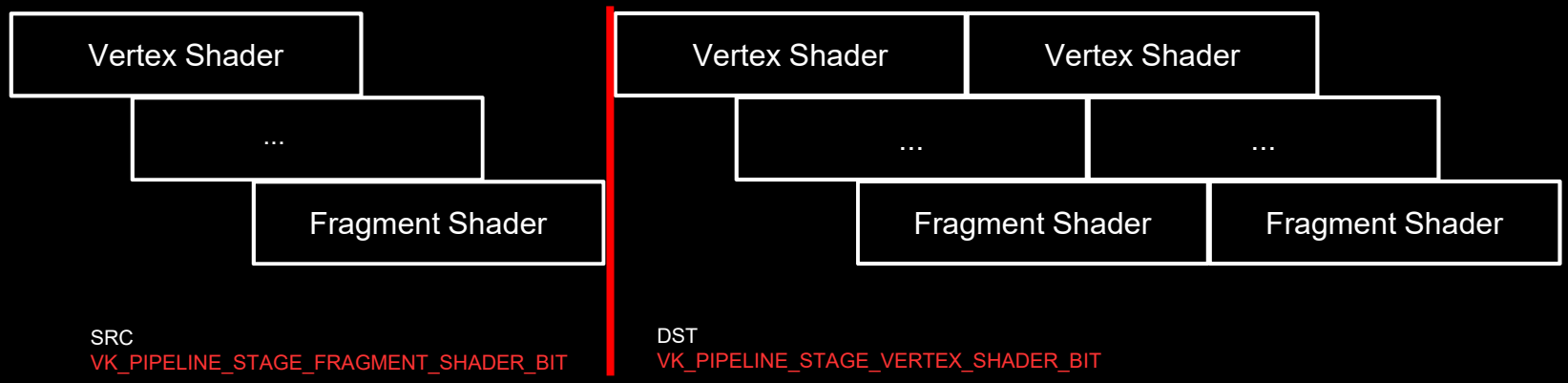
- Pipeline Barriers
  - Ensure most optimal stage masks
  - Batch them!
- Load/Store Ops
  - You don’t always have to care!
  - LoadOpLoad and StoreOpStore
- Cache Stuff
  - PSOs
  - Descriptor Sets

General Vulkan Optimizations

- Pipeline Barriers

— Ensure optimal use of stage masks

— This is NOT optimal

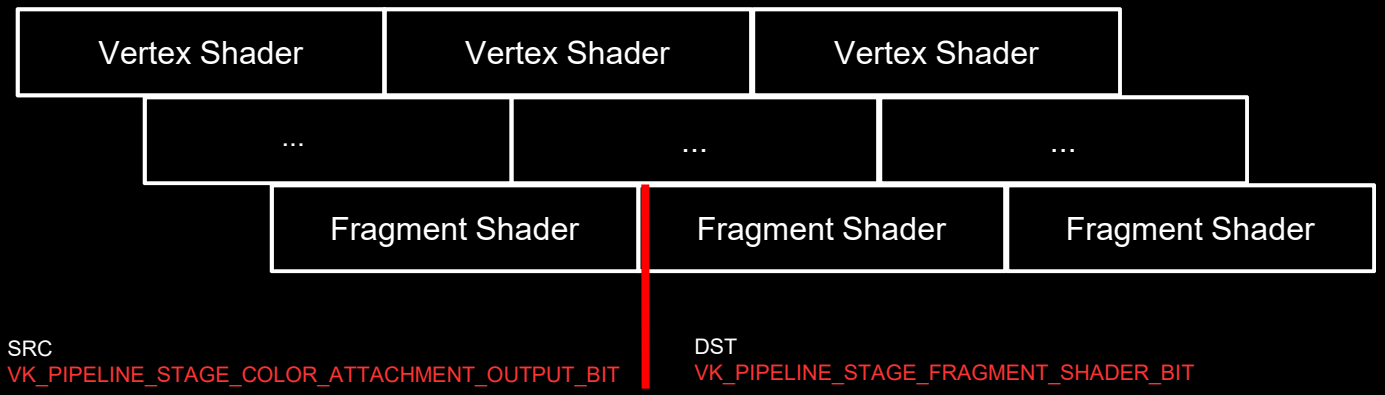


General Vulkan Optimizations

- Pipeline Barriers

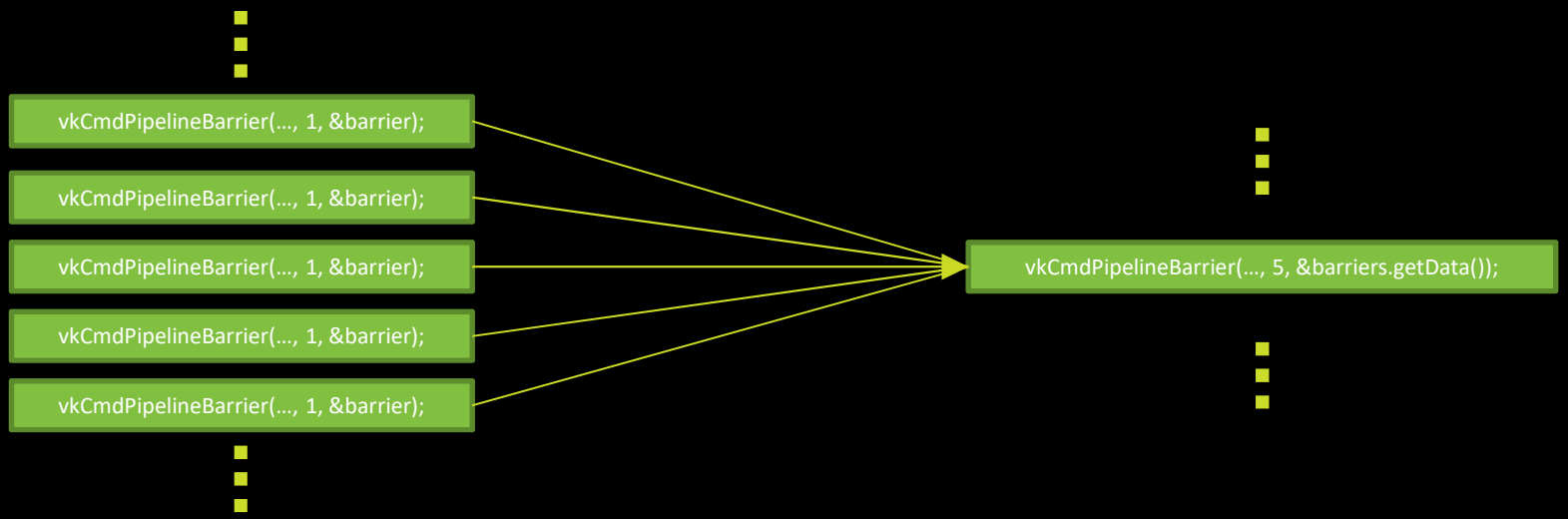
— Ensure optimal use of stage masks

— This IS optimal



General Vulkan Optimizations

- Pipeline Barriers
  - Batch them!



General Vulkan Optimizations

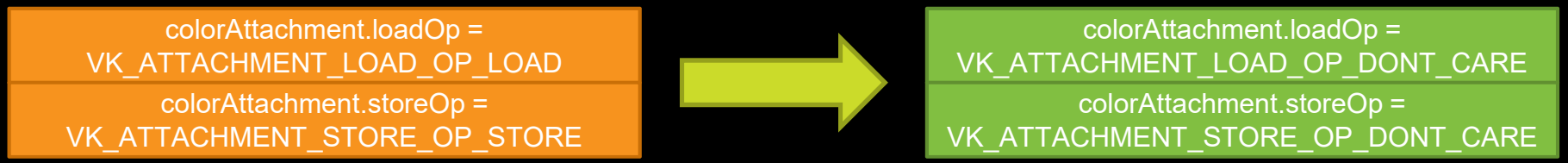
- Pipeline Barriers
  - Ensure most optimal stage masks
  - Batch them!
- Load/Store Ops
  - You don't always have to care!
  - LoadOpLoad and StoreOpStore
- Cache Stuff
  - PSOs
  - Descriptor Sets

Up to 40% performance

General Vulkan Optimizations

- Load/Store Ops

— You don’t always have to care!



General Vulkan Optimizations

- Pipeline Barriers
  - Ensure most optimal stage masks
  - Batch them!
- Load/Store Ops
  - You don't always have to care!
  - LoadOpLoad and StoreOpStore
- Cache Stuff
  - Use VKPipelineCache
  - Descriptor Sets

Up to 40% performance

Up to 50% bandwidth

General Vulkan Optimizations

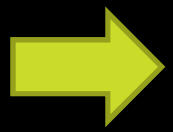
- Cache Stuff
  - Use VkPipelineCache
  - Descriptor Sets

1st Run

```
vkCreateGraphicsPipelines(device, VK_NULL_HANDLE, …);
```

2nd Run

```
vkCreateGraphicsPipelines(device, VK_NULL_HANDLE, …);
```



1st Run

```
vkCreateGraphicsPipelines(device, VK_NULL_HANDLE, …);
vkCreatePipelineCache(device, &createInfo, NULL, &cache);
```

2nd Run

```
vkCreateGraphicsPipelines(device, pipelineCache, …);
```

General Vulkan Optimizations

- Cache Stuff

— Use VkPipelineCache

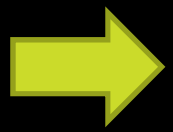
— Descriptor Sets

1st Frame

```
vkUpdateDescriptorSets(device, …);
```

All remaining Frames

```
vkUpdateDescriptorSets(device, …);
```



1st Frame

```
vkUpdateDescriptorSets(device, …);
someDescriptorMap.emplace(resourceKey, &descriptorSet);
```

All remaining Frames

```
descriptorSet = someDescriptorMap.find(resourceKey);
```

General Vulkan Optimizations

- Pipeline Barriers
  - Ensure most optimal stage masks
  - Batch them!
- Load/Store Ops
  - You don't always have to care!
  - LoadOpLoad and StoreOpStore
- Cache Stuff
  - Reuse
  - Descriptor Sets

Up to 40% performance

Up to 50% bandwidth

5-10% CPU time

General Vulkan Optimizations

- WaitForFence
  - Try to avoid it!
  - Use a pool of Command Buffers
- RenderPasses
  - Minimize switching
  - Combine passes!
- Threads
  - Put stuff on other threads.

General Vulkan Optimizations

- WaitForFence
  - Try to avoid it!
  - Use a pool of Command Buffers
- RenderPasses
  - Minimize switching
  - Combine passes!
- Threads
  - Put stuff on other threads.

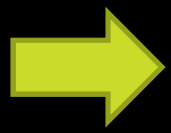
General Vulkan Optimizations

- WaitForFences

— Try to avoid it!

— Make use of pools of command buffers

```
…
vkWaitForFences(device, 1, &fence, true, timeout);
…
```



```
result = vkGetFenceStatus(device, fence);
if(result == VK_STATUS_NOT_READY)
fencePool.push_back(fence);
```

```
if(!fencePool.empty())
fence = fencePool.pop_back();
Else
fence = vkCreateFence(…, &fenceToUse);
```

General Vulkan Optimizations

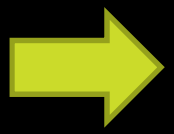
- WaitForFence
  - Try to avoid it!
  - Use a pool of Command Buffers
- RenderPasses
  - Minimize switching
  - Combine passes!
- Threads
  - Put stuff on other threads.

General Vulkan Optimizations

- RenderPasses
  - Minimize switching

```
vkCmdBeginRenderPass(commandBuffer, &beginInfo, contents);
DrawObjects1();
vkCmdEndRenderPass(commandBuffer);
```

```
vkCmdBeginRenderPass(commandBuffer, &beginInfo, contents);
DrawObjects2();
vkCmdEndRenderPass(commandBuffer);
```



```
vkCmdBeginRenderPass(commandBuffer, &beginInfo, contents);
DrawObjects1();
DrawObjects2();
vkCmdEndRenderPass(commandBuffer);
```

General Vulkan Optimizations

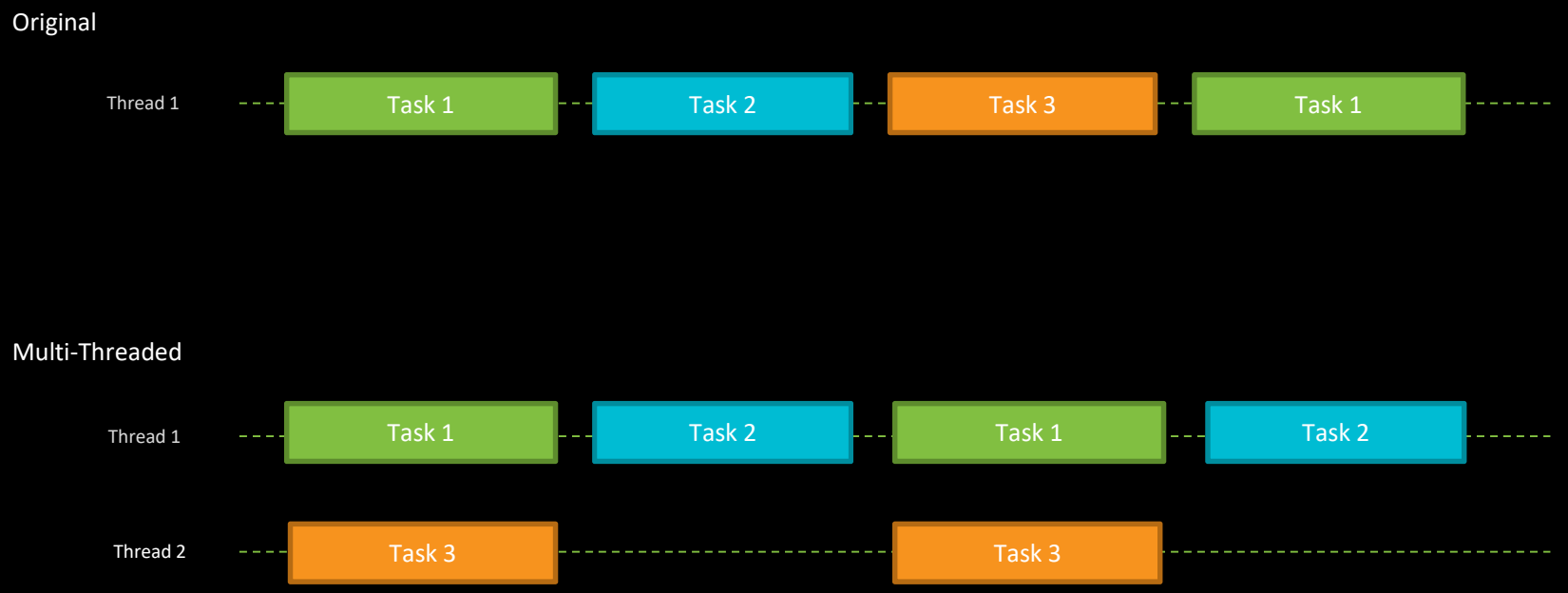
- WaitForFence
  - Try to avoid it!
  - Use a pool of Command Buffers
- RenderPasses
  - Minimize switching
  - Combine passes!
- Threads
  - Put stuff on other threads.

Up to 10% Performance

1-5% Performance

General Vulkan Optimizations

- Threads



General Vulkan Optimizations

- WaitForFence
  - Try to avoid it!
  - Use a pool of Command Buffers
- RenderPasses
  - Minimize switching
  - Combine passes!
- Threads
  - Put stuff on other threads.

Up to 10% Performance

1-5% Performance

Up to 30% Performance

Vulkan Optimizations in Call of Duty Mobile

- Single Scratch Buffer
- Low Priority Destroy Thread
- Descriptor Set Batching/Caching
- RenderPass Load/Store
- Threaded Present
- VB, IB Binding Optimization
- Direct Buffer Access

Vulkan Optimizations

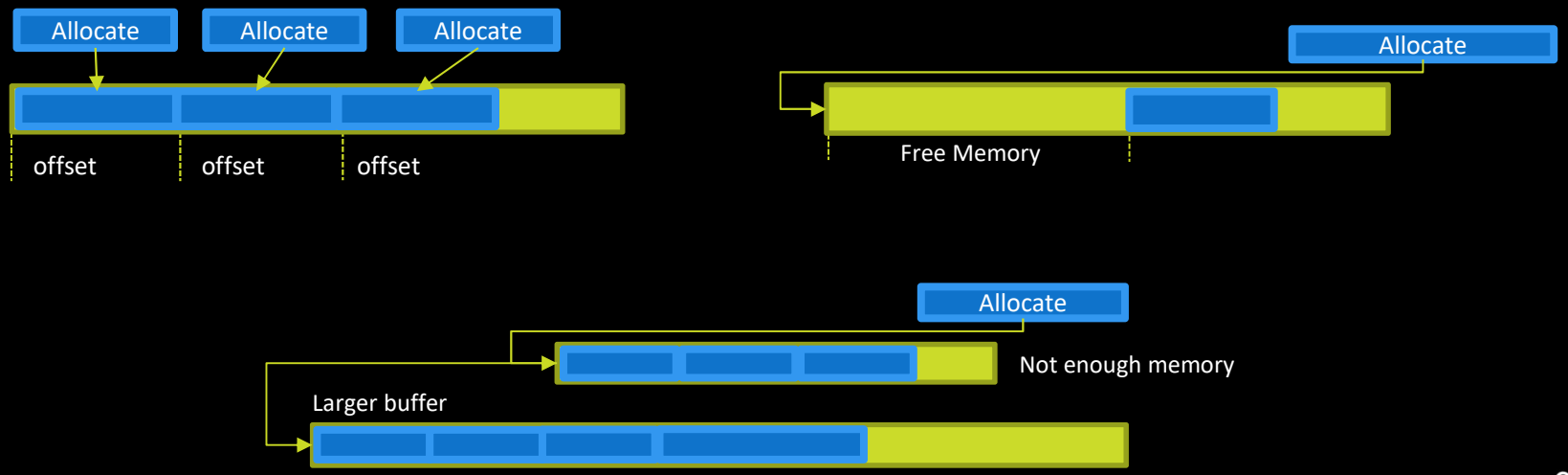
- Single Scratch Buffer
- Low Priority Destroy Thread
- Descriptor Set Batching/Caching
- RenderPass Load/Store
- Threaded Present
- VB, IB Binding Optimization
- Direct Buffer Access

Vulkan Optimizations

- Single Scratch Buffer

— Ring buffer for all needs

— Usually recreated only during first load

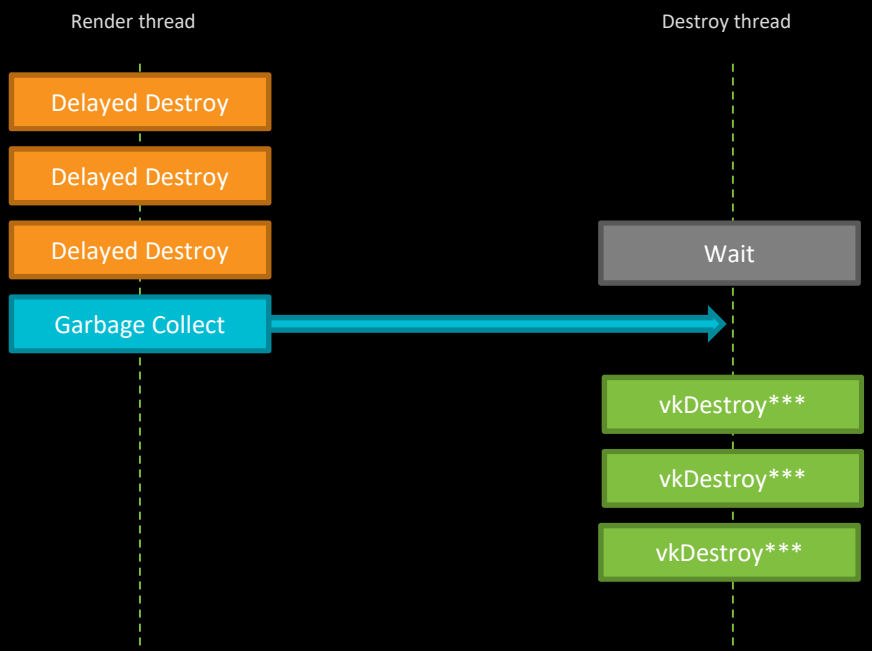


Vulkan Optimizations

- Single Scratch Buffer
- Low Priority Destroy Thread
- Descriptor Set Batching/Caching
- RenderPass Load/Store
- Threaded Present
- VB, IB Binding Optimization
- Direct Buffer Access

Vulkan Optimizations

- Low Priority Destroy Thread



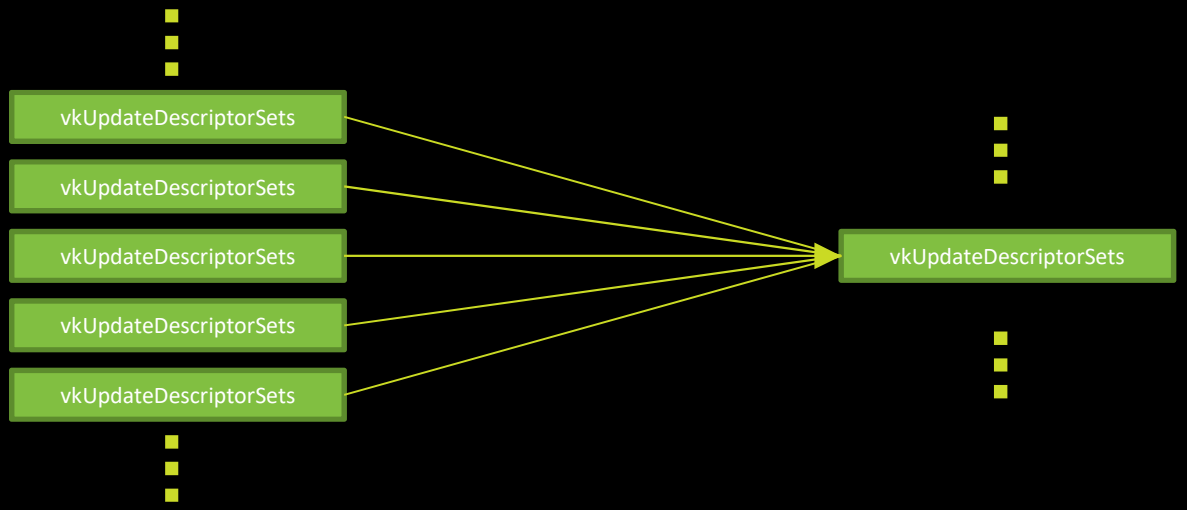
Vulkan Optimizations

- Single Scratch Buffer
- Low Priority Destroy Thread
- Descriptor Set Batching/Caching
- RenderPass Load/Store
- Threaded Present
- VB, IB Binding Optimization
- Direct Buffer Access

Vulkan Optimizations

- Descriptor Set Batching/Caching

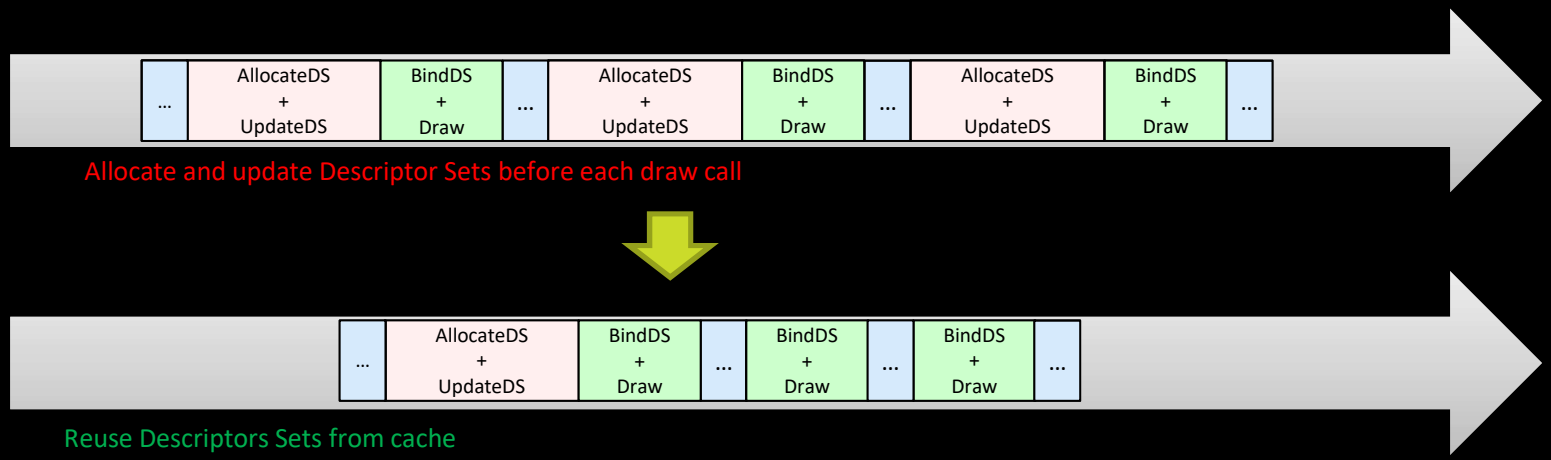
— Combine separate Update calls into one



Vulkan Optimizations

- Descriptor Set Batching/Caching

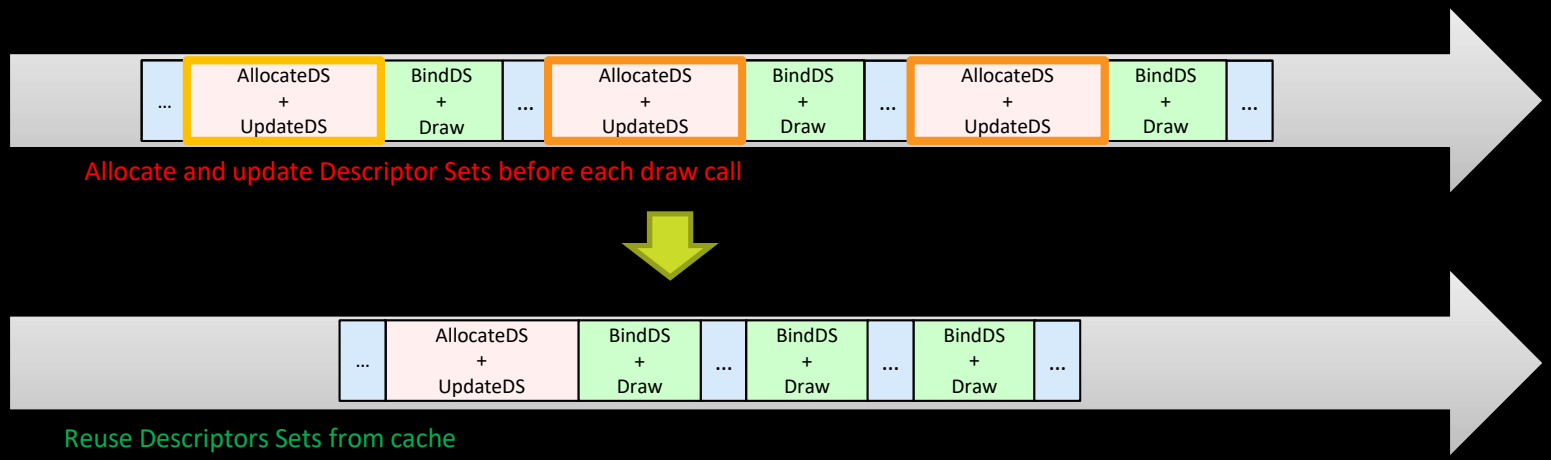
— Cache and reuse sets with same resources



Vulkan Optimizations

- Descriptor Set Batching/Caching

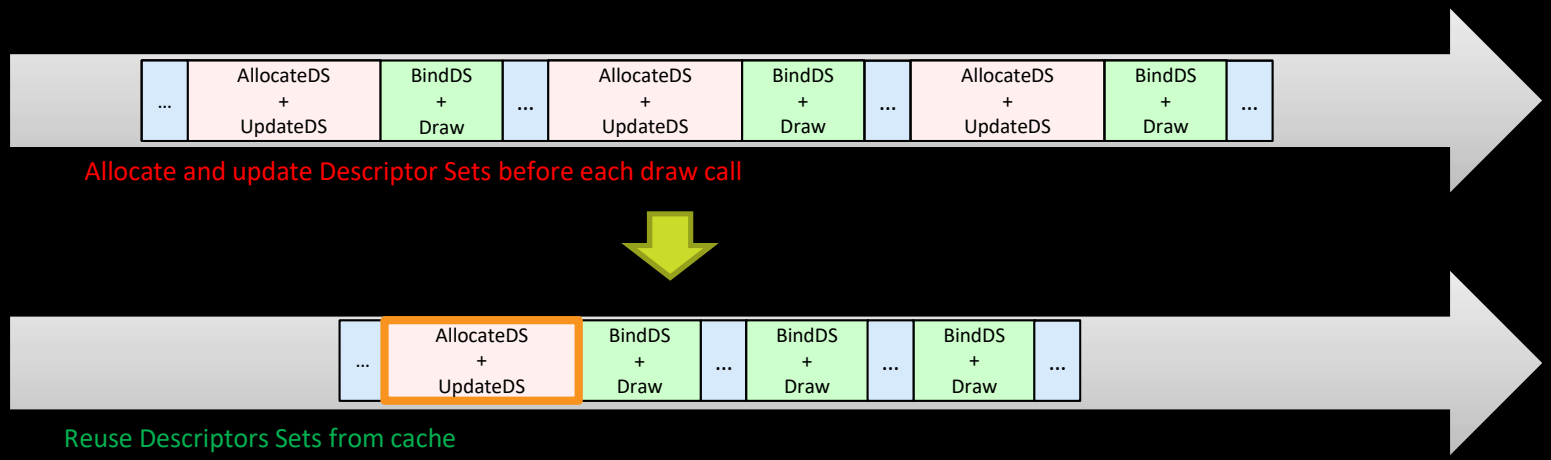
— Cache and reuse sets with same resources



Vulkan Optimizations

- Descriptor Set Batching/Caching

— Cache and reuse sets with same resources



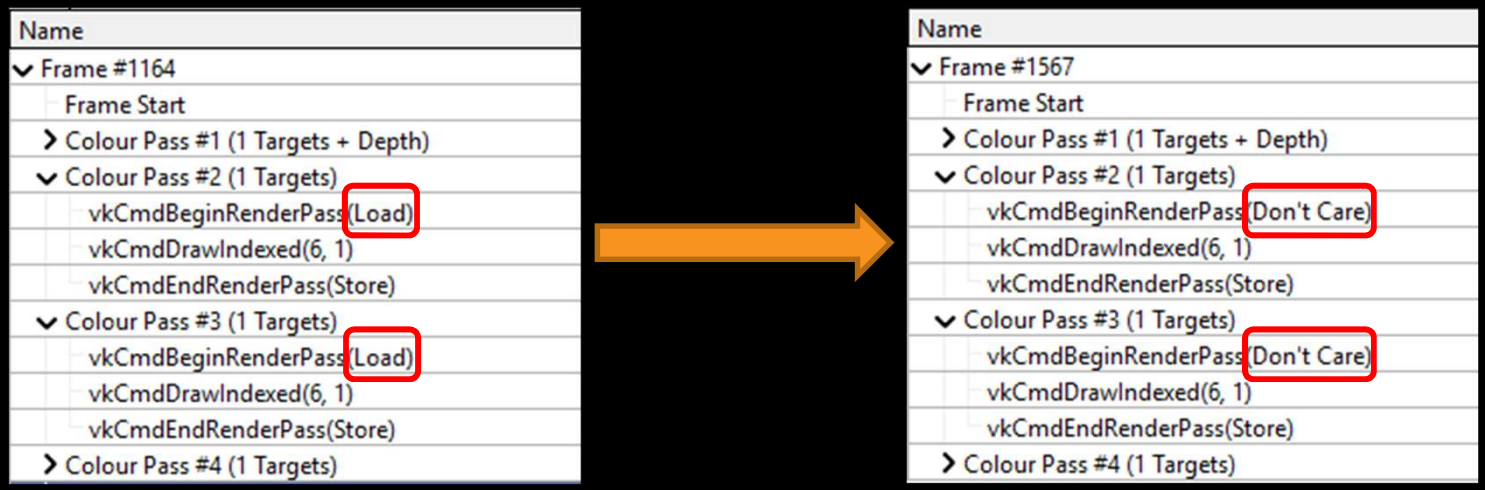
Vulkan Optimizations

- Single Scratch Buffer
- Low Priority Destroy Thread
- Descriptor Set Batching/Caching
- RenderPass Load/Store
- Threaded Present
- VB, IB Binding Optimization
- Direct Buffer Access

Vulkan Optimizations

- RenderPass Load/Store

  When render target is being fully redrawn there is no need to load previous data.

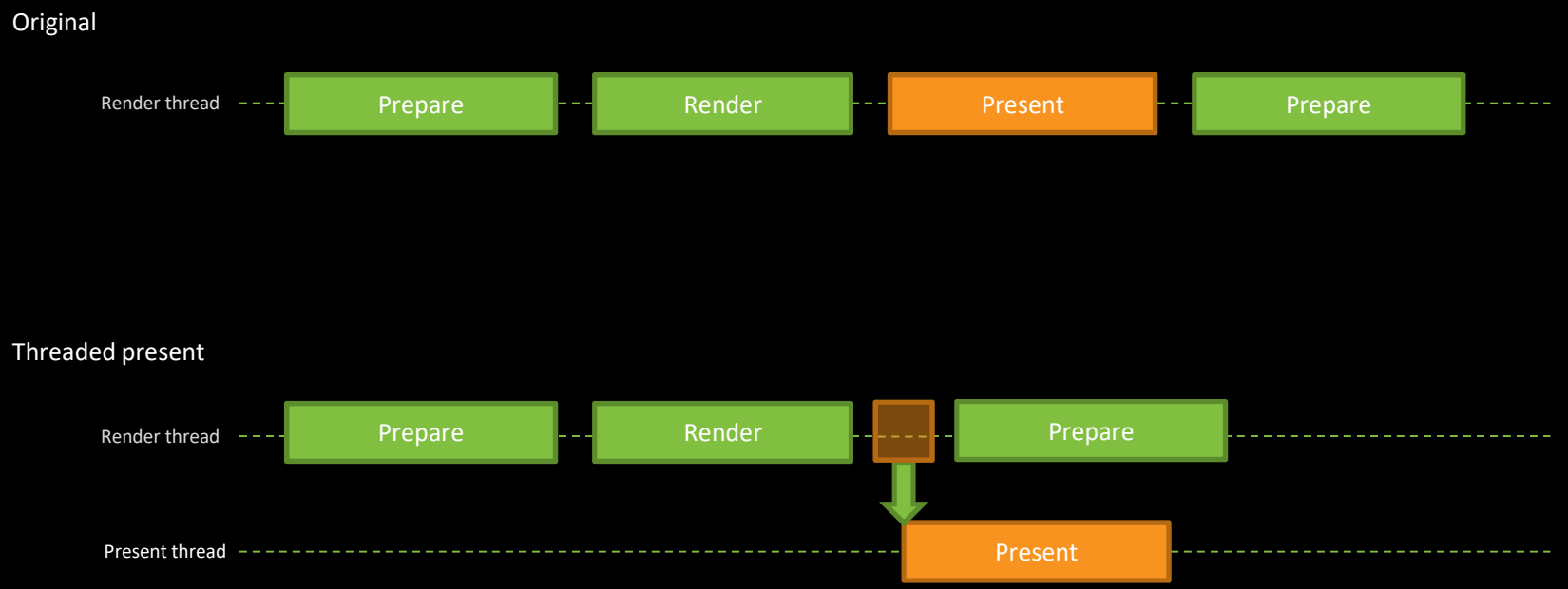


Vulkan Optimizations

- Single Scratch Buffer
- Low Priority Destroy Thread
- Descriptor Set Batching/Caching
- RenderPass Load/Store
- Threaded Present
- VB, IB Binding Optimization
- Direct Buffer Access

Vulkan Optimizations

- Threaded Present



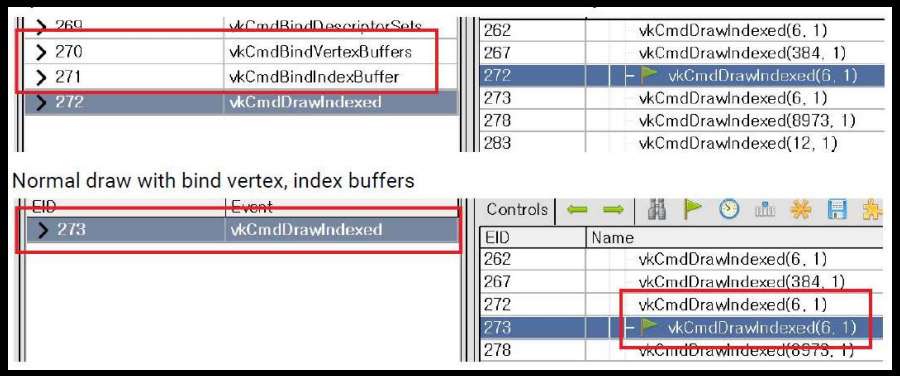
Vulkan Optimizations

- Single Scratch Buffer
- Low Priority Destroy Thread
- Descriptor Set Batching/Caching
- RenderPass Load/Store
- Threaded Present
- VB, IB Binding Optimization
- Direct Buffer Access

Vulkan Optimizations

- VB, IB Binding Optimization

  As buffers are same there is no need to bind them every draw call.

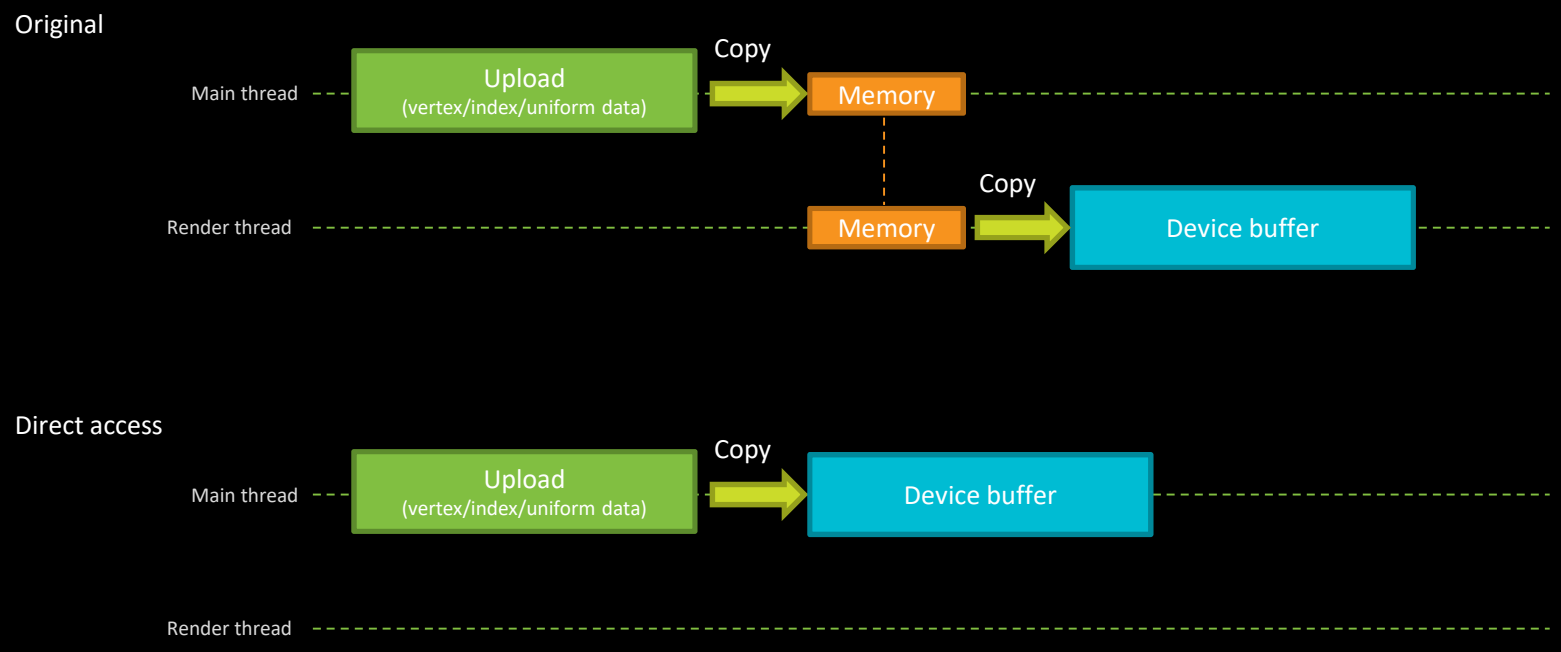


Vulkan Optimizations

- Single Scratch Buffer
- Low Priority Destroy Thread
- Descriptor Set Batching/Caching
- RenderPass Load/Store
- Threaded Present
- VB, IB Binding Optimization
- Direct Buffer Access

Vulkan Optimizations

- Direct Buffer Access




CALL OF DUTY
MOBILE
ACTIVISION
腾讯游戏
Tencent Games
SAMSUNG
unity
Vulkan
Adaptive Performance

Top 3 Takeaways

1: Use Adaptive Performance!

gamedev@samsung.com

Top 3 Takeaways

1: Use Adaptive Performance!

2: Check your barriers!

gamedev@samsung.com

Top 3 Takeaways

1: Use Adaptive Performance!

2: Check your barriers!

3: Check your load/store ops!

gamedev@samsung.com

SAMSUNG

Galaxy GameDev

Top 3 Takeaways

1: Use Adaptive Performance!

2: Check your barriers!

3: Check your load/store ops!

3.5: Thread stuff!

gamedev@samsung.com

Thank You!

gamedev@samsung.com

benjamin.m@samsung.com